ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ

Кафедра
«Криптология и кибербезопасность»

Лабораторная работа №4

по предмету «Технологии контейнеризации»

Выполнил студент группы Б20-505

Сорочан Илья

Москва – 2023

Содержание

1. Vagrant

В качестве хоста для контейнера wireguard будет использоваться виртуальная машина `ubuntu/jammy64`.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

$docker = <<-SCRIPT
# Update and install requirments
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update

# Install docker-engine
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
SCRIPT

Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/jammy64"

  config.vm.box_check_update = false

  config.vm.provision "shell", inline: $docker, run: "once"
end
```

Рис. 1. Начальная конфигурация виртуальной машины

В первую очередь так как мы используем wireguard, то необходимо настроить сеть следующим образом:

```
    config.vm.network "forwarded_port", guest: 51820, host: 51820
    config.vm.network "forwarded_port", guest: 10086, host: 10086

    config.vm.network "public_network"
```

Рис. 2. Настройка сети виртуальной машины

Здесь стоит отметить несколько моментов:

- порт 51820 используется для подключений wireguard;
- порт 10086 используется wgdashboard для web-интерфейса;
- используется «public_network» так как необходим доступ в сеть изнутри машины и возможность доступа с нашей машины к виртуальной по сети.

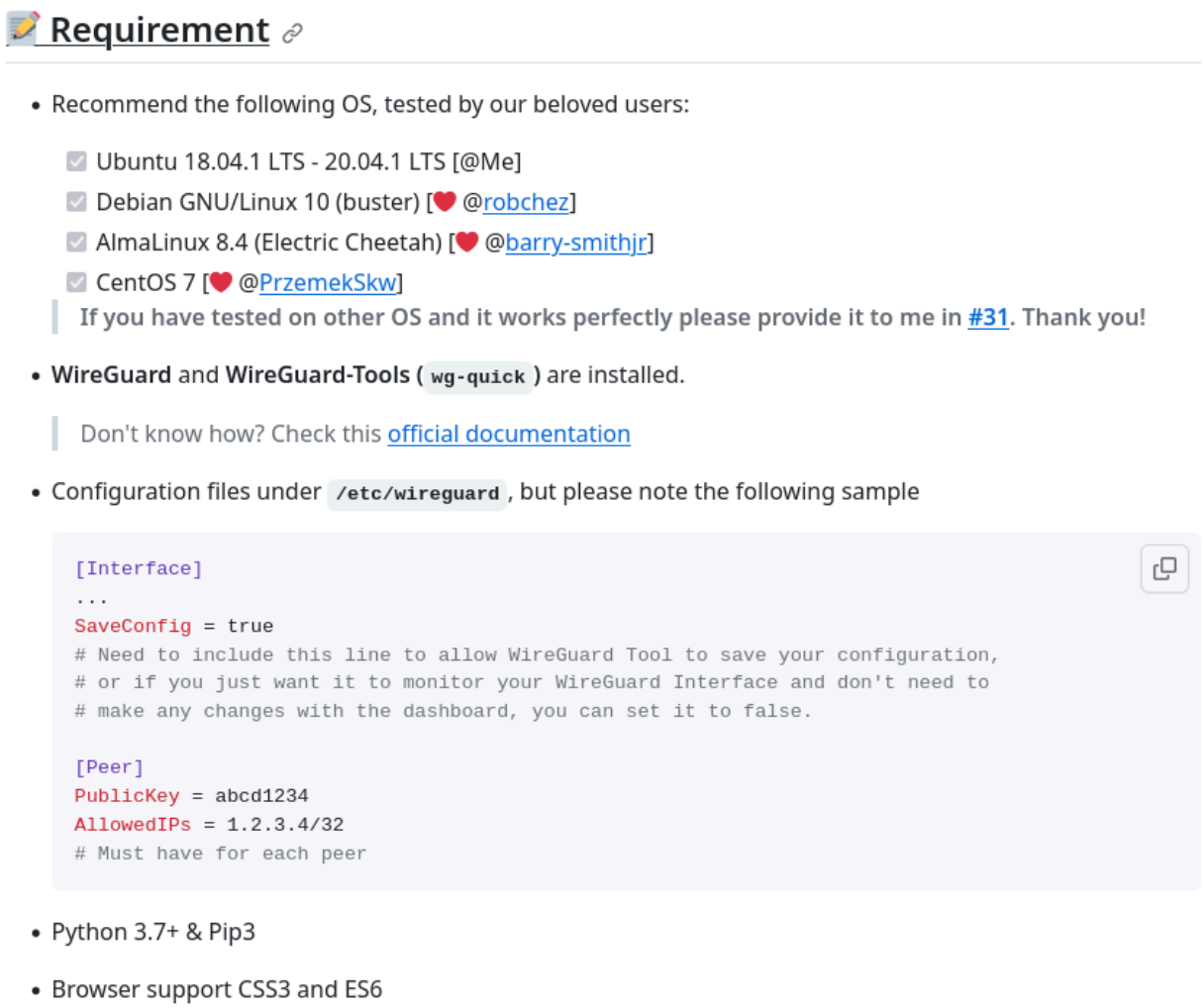
На последок стоит добавить копирование файлов `Dockerfile` и `docker-compose.yaml`:

```
   config.vm.provision "file", source: "Dockerfile", destination: "Dockerfile"
   config.vm.provision "file", source: "docker-compose.yml", destination: "docker-compose.yml"
```

Рис. 3. Копирование Dockerfile и docker-compose.yml

2. Dockerfile

Предже всего следует отметить, что наиболее популярный контейнер `linuxserver/wireguard` использует alpine linux, которого нет среди протестированных дистрибутивов для wgdashboard:



📝 Requirement

- Recommend the following OS, tested by our beloved users:
  - Ubuntu 18.04.1 LTS - 20.04.1 LTS [@Me]
  - Debian GNU/Linux 10 (buster) [❤️ @robchez]
  - AlmaLinux 8.4 (Electric Cheetah) [❤️ @barry-smithjr]
  - CentOS 7 [❤️ @PrzemekSkw]

  > If you have tested on other OS and it works perfectly please provide it to me in #31. Thank you!

- **WireGuard** and **WireGuard-Tools** (`wg-quick`) are installed.

  > Don't know how? Check this official documentation

- Configuration files under `/etc/wireguard`, but please note the following sample

```
[Interface]
...
SaveConfig = true
# Need to include this line to allow WireGuard Tool to save your configuration,
# or if you just want it to monitor your WireGuard Interface and don't need to
# make any changes with the dashboard, you can set it to false.

[Peer]
PublicKey = abcd1234
AllowedIPs = 1.2.3.4/32
# Must have for each peer
```

- Python 3.7+ & Pip3
- Browser support CSS3 and ES6

Рис. 4. Необходимые требования wg-dashboard

Одним из вариантов является использование данного контейнера и доработка его до совместимости с wgdashboard. Однако в данной лабораторной был выбран другой путь: использовать убунту (`ubuntu:20.04`), на которую затем установить сначала wireguard, затем wgdashboard.

2.1. Установка необходимых пакетов

```
FROM ubuntu:20.04

# Update
RUN apt-get update && \
    apt-get install -y wireguard iproute2 net-tools git python3-pip gunicorn  && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

```

Рис. 5. Установка необходимых пакетов

Среди них:

- wireguard
- iproute2 - устанавливается ради утилиты ip
- net-tools - ifconfig

А затем производится чистка.

2.2. Генерация конфигурации wireguard

```
# Setup wireguard config
RUN echo -n "[Interface]\nPrivateKey = " > /etc/wireguard/wg0.conf && \
    wg genkey | tee -a /etc/wireguard/wg0.conf | wg pubkey > publickey && \
    echo -n "Address = 10.0.0.1/" >> /etc/wireguard/wg0.conf && \
    ip -o -f inet addr show eth0 | awk '{split($4, a, "/"); print a[2]}' >> /etc/wireguard/wg0.conf && \
    echo "ListenPort = 51820" >> /etc/wireguard/wg0.conf && \
    echo -n "\n[Peer]\nPublicKey = " >> /etc/wireguard/wg0.conf && \
    cat publickey >> /etc/wireguard/wg0.conf && rm -f publickey && \
    echo "AllowedIPs = 0.0.0.0/0" >> /etc/wireguard/wg0.conf

```

Рис. 6. Генерация конфигурации wireguard

Для базовой работы wireguard была сгенерирована конфигурация сервера wireguard в автоматическом режиме.

2.3. Установка wgdashboard

```
# Setup wg-dashboard
RUN cd /usr/local/share && \
    git clone -b v3.0.6 https://github.com/donaldzou/WGDashboard.git wgdashboard && \
    cd wgdashboard/src && \
    chmod u+x wgd.sh && ./wgd.sh install && \
    sudo chmod -R 755 /etc/wireguard

```

Рис. 7. Установка wgdashboard согласно инструкции

2.4. Открытие портов

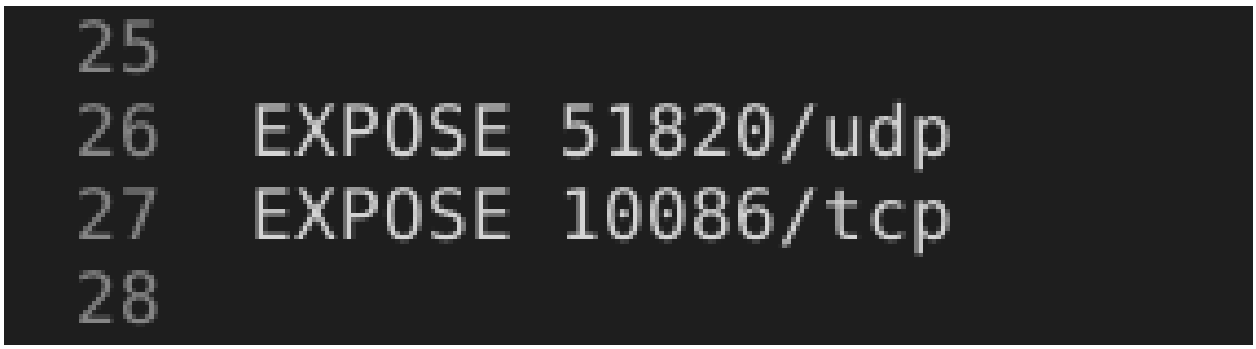


Рис. 8. Открытие портов для использования с `-P`

2.5. `Entrypoint` контейнера

Рис. 9. `Entrypoint` контейнера

Состоит из следующих частей:

- `wg-quick up wg0` - запуск wireguard с конфигурацией, сгененрированной ранее;
- `cd /usr/local/share/wgdashboard/src && ./wgd.sh start` - запукс wgdashboard согласно инструкции;
- `tail -f /dev/null` - так как оба предыдущих процесса работают «в фоне» необходимо создать основной процесс.

2.6. Сборка и запуск

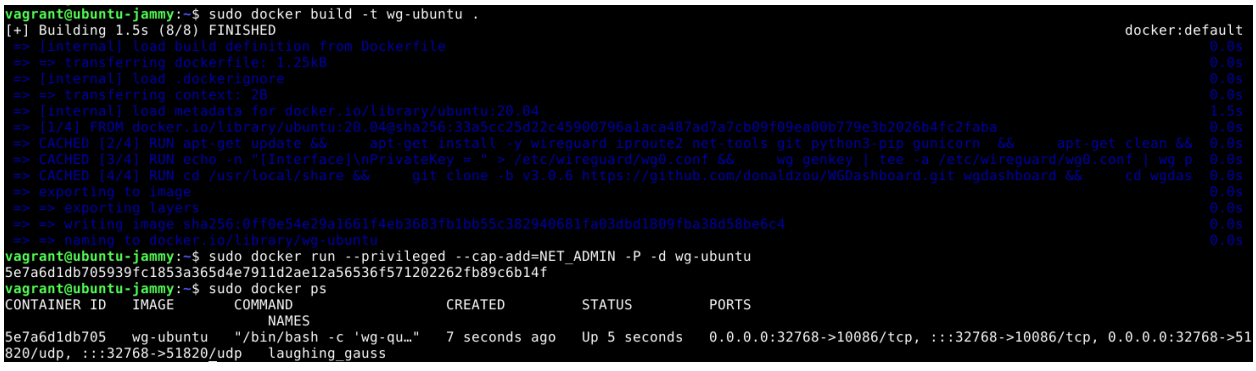


Рис. 10. Сборка и запуск контейнера

2.7. Метаданные

```
2
3  LABEL author="Sorochan I.V."
4  LABEL email="none"
5  LABEL version="0.1.0"
6  LABEL description="wgdashboard + wireguard + ubuntu"
7
```

Рис. 11. Метаданные

2.8. Рабочая директория

```
7
8  WORKDIR /app
9
```

Рис. 12. Рабочая директория

2.9. Создание пользователя

```
15
16 RUN adduser user01 --disabled-password && \
17     echo "user01 ALL=(ALL) NOPASSWD: ALL" >> /etc/sudoers
18
```

Рис. 13. Создание пользователя user01

3. Docker compose

3.1. Базовый образ

```
version: '3'
services:
  wg-ubuntu:
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - 51820:51820/udp
      - 10086:10086/tcp
    entrypoint: bash -c "wg-quick up wg0 && cd /usr/local/share/wgdashboard/src && ./wgd.sh start && tail -f /dev/null"
~
```

Рис. 14. Файл docker-compose.yml

Здесь мы используем собранный образ из существующего Dockerfile.

```
vagrant@ubuntu-jammy:~$ sudo docker compose up -d
[+] Running 2/2
 ✔ Network vagrant_default          Created
 ✔ Container vagrant-wg-ubuntu-1    Started
vagrant@ubuntu-jammy:~$ sudo docker compose logs
vagrant-wg-ubuntu-1  | Warning: `/etc/wireguard/wg0.conf' is world accessible
vagrant-wg-ubuntu-1  | [#] ip link add wg0 type wireguard
vagrant-wg-ubuntu-1  | [#] wg setconf wg0 /dev/fd/63
vagrant-wg-ubuntu-1  | [#] ip -4 address add 10.0.0.1/16 dev wg0
vagrant-wg-ubuntu-1  | [#] ip link set mtu 1420 up dev wg0
vagrant-wg-ubuntu-1  | ------------------------------------------------------------
vagrant-wg-ubuntu-1  | | Starting WGDashboard with Gunicorn in the background.    |
vagrant-wg-ubuntu-1  | | Log files is under log/                                  |
vagrant-wg-ubuntu-1  | ------------------------------------------------------------
```

Рис. 15. Запуск docker compose up

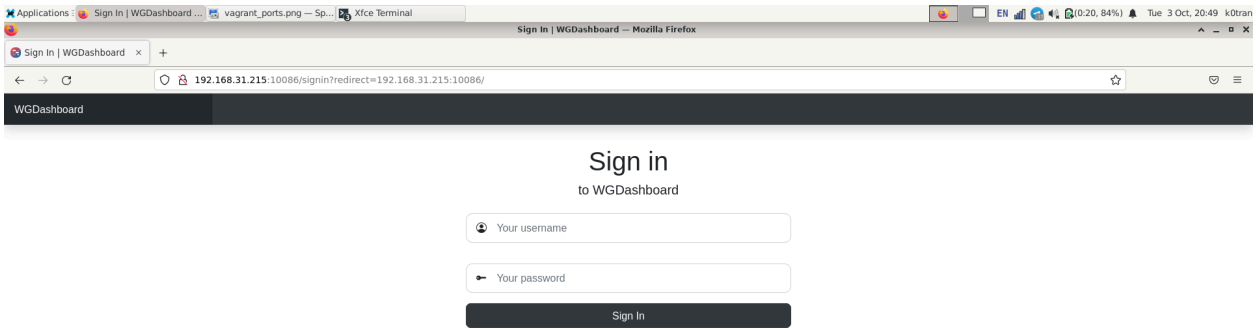


Рис. 16. Веб форма wgdashboard

3.2. Сети и тома

Рис. 17. Том

Рис. 18. Сетки 1

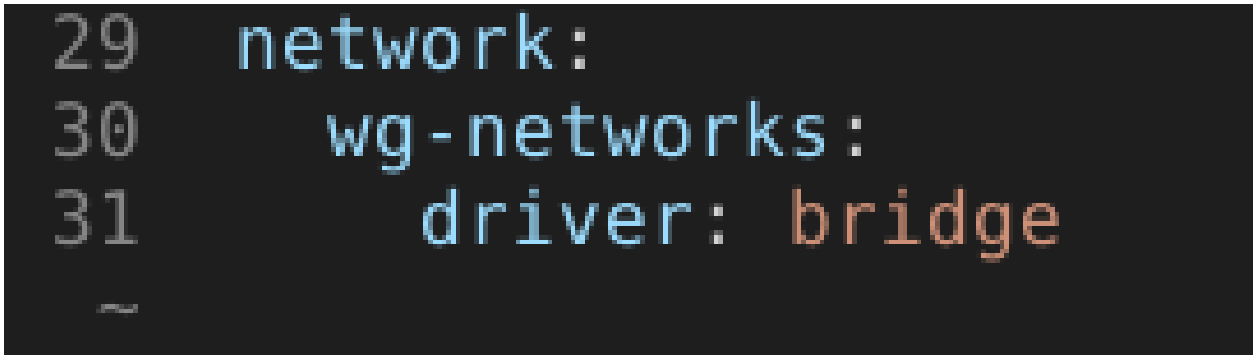


Рис. 19. Сетки 2

3.3. Директива restart

Рис. 20. Restart

Виды restart:

- no
- on-failure
- always
- unless-stopped

3.4. Ограничения CPU и RAM

```
20        deploy:
21          resources:
22            limits:
23              cpus: '0.5'
24              memory: 512M
25            reservations:
26              cpus: '0.2'
27              memory: 256M
28
```

Рис. 21. Ограничения CPU и RAM

3.5. Второй запуск

```
vagrant@ubuntu-jammy:~$ sudo docker compose up -d --force-recreate
[+] Running 2/2
 ✔ Network vagrant_wg-networks    Created                                                    0.1s
 ✔ Container vagrant-wg-ubuntu-1  Started                                                    0.0s
vagrant@ubuntu-jammy:~$ sudo docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED         STATUS         PORTS
                                                                          NAMES
fad6238c6c6a   vagrant-wg-ubuntu  "bash -c 'wg-quick u…"   6 seconds ago   Up 6 seconds   0.0.0.0:10086->10086/
tcp, :::10086->10086/tcp, 0.0.0.0:51820->51820/udp, :::51820->51820/udp   vagrant-wg-ubuntu-1
vagrant@ubuntu-jammy:~$
```

Рис. 22. Ограничения CPU и RAM